21 Analysis of Benchmarks

In order to assess strengths and weaknesses of different semi-supervised learning (SSL) algorithms, we invited the chapter authors to apply their algorithms to eight benchmark data sets. These data sets encompass both artificial and real-world problems. We provide details on how the algorithms were applied, especially how hyperparameters were chosen given the few labeled points. Finally, we present and discuss the empirical performance.

21.1 The Benchmark

21.1.1 Data Sets

The benchmark consists of eight data sets as shown in table 21.1. Three of them were artificially created in order to create situations that correspond to certain assumptions (cf. chapter 1); this was done to allow for relating the performance of the algorithms to those assumptions. The five other benchmark data sets were derived from real data. It can thus be hoped that the performance on these is indicative of the performance in real applications.

Table 21.1 Basic properties of benchmark data sets

Data set	**Classes**	**Dimension**	**Points**	**Comment**
`g241c`	2	241	1500	artificial
`g241d`	2	241	1500	artificial
`Digit1`	2	241	1500	artificial
`USPS`	2	241	1500	imbalanced
`COIL`	6	241	1500	
`BCI`	2	117	400	
`Text`	2	11,960	1500	sparse discrete
`SecStr`	2	315	83,679	sparse binary

The purpose of the benchmark was to evaluate the power of the presented algorithms themselves in a way as neutral as possible. Thus ideally the data preprocessing should be similar for all algorithms; in particular, it should be avoided that in some cases it takes advantage of domain knowledge, when in others it does not. To prevent the experimenters from using domain knowledge, we tried to obscure structure in the data (e.g. by shuffling the pixels in the images), and even to hide the identity of the data sets (e.g. by also shuffling the data points). Also, we used the same number of dimensions (241) and points (1500) for most data sets in the same attempt to obscure the origin of the data and in order to increase the comparability of the results. However, we did provide information as to which data sets originate from images and which from text.

All data sets are available for further research at `http://www.kyb.tuebingen.mpg.de/ssl-book/`.

g241c This data set was generated such that the cluster assumption holds, i.e. the classes correspond to clusters, but the manifold assumption does not. First, 750 points were drawn from each of two unit-variance isotropic Gaussians (i.e., from $\mathcal{N}(\boldsymbol{\mu}_i, \mathbf{I})$), the centers of which had a distance of 2.5 in a random direction (i.e., $\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\| = 2.5$). The class label of a point represents the Gaussian it was drawn from. Finally, all dimensions are standardized, i.e. shifted and rescaled to zero-mean and unit variance. A two-dimensional projection of the data is shown on the left side of figure 21.1.

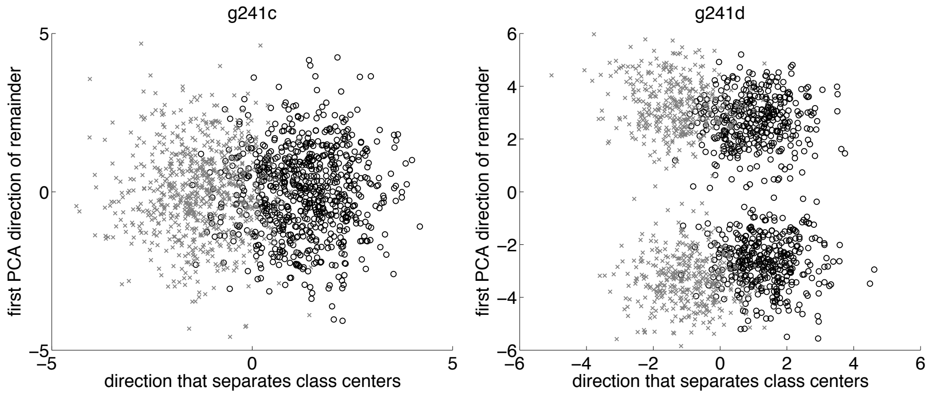

Figure 21.1 Two-dimensional projections of `g241c` (*left*) and `g241d` (*right*). Black circles, class +1; gray crosses, class -1.

g241d This data set was constructed to have potentially misleading cluster structure, and no manifold structure. First 375 points were drawn from each of two unit-variance isotropic Gaussians, the centers of which have a distance of 6 in a random direction; these points form the class +1. Then the centers of two further Gaussians for class -1 were fixed by moving from each of the former centers a distance of 2.5 in a random direction. Again, the identity matrix was used as covariance matrix, and 375 points were sampled from each new Gaussian. A two-dimensional projection of the resulting data is shown on the right side of figure 21.1.

Digit1 This data set was designed to consist of points close to a low-dimensional manifold embedded into a high-dimensional space, but not to show a pronounced cluster structure. We therefore started from a system that generates artificial writings (images) of the digit "1" developed by Matthias Hein (Hein and Audibert, 2005). The images are constructed starting from an abstract "1" implemented as a function $[0,1]^2 \rightarrow \{0,1\}$, with the main vertical line ranging from $y = 0.2$ to $y = 0.8$ at $x = 0.5$. There are five degrees of freedom in this function: two for translations ($[-0.13, +0.13]$ each), one for rotation ($[-90°, +90°]$), one for line thickness ($[0.02, 0.05]$), and one for the length of a small line at the bottom ($[0, 0.1]$). The resulting function is then discretized to an image of size 16×16. As an example, the first data point is shown in figure 21.2 (left).

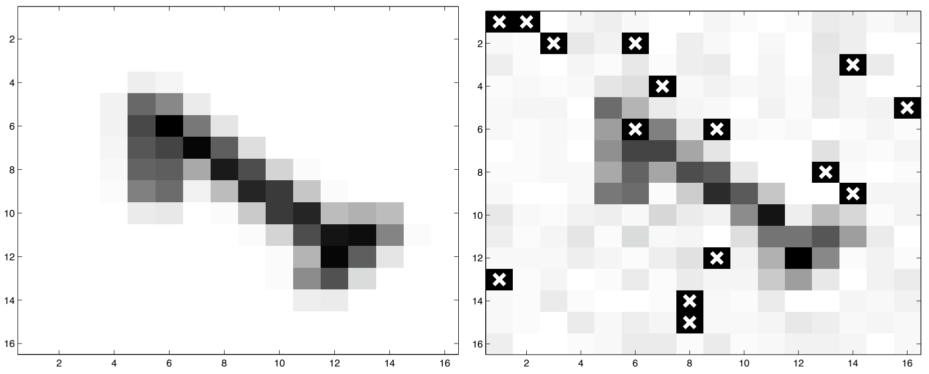

Figure 21.2 First data point from `Digit1` data set. (*Left*) Original image. (*Right*) After rescaling, adding noise, and masking dimensions (x).

We randomly sampled 1500 such images. The class label was set according to the tilt angle, with the boundary corresponding to an upright digit. To make the task a bit more difficult, we apply a sequence of transformations to the data as shown in algorithm 21.1, with σ set to 0.05. The result of this transformation (except for bias and permutation) applied to the first data point is shown in the right part of figure 21.2.

Since the data lie close to a five-dimensional manifold, SSL methods based on the manifold assumption are expected to improve substantially on supervised learning.

Algorithm 21.1 Obscure image data

```
Require: $\sigma$ {standard deviation of random noise}
1: randomly select and permute 241 columns (features)
2: add to each column a random bias drawn from $\mathcal{N}(0, 1)$
3: multiply each column by a value from unif($[-1, -0.5] \cup [0.5, 1]$)
4: add independent noise from $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ to each row (data point)
```

USPS We derived a benchmark data set from the famous USPS set of handwritten digits as follows. We randomly drew 150 images of each of the ten digits. The digits "2" and "5" were assigned to the class +1, and all the others formed class −1. The classes are thus imbalanced with relative sizes of 1:4. We also expect both the cluster assumption and the manifold assumption to hold.

To prevent people from realizing the origin of this benchmark data set and exploiting its known structure (e.g. the spatial relationship of features in the image), we again obscured the data by application of algorithm 21.1, this time with $\sigma = 0.1$. Figure 21.3 illustrates the impact.

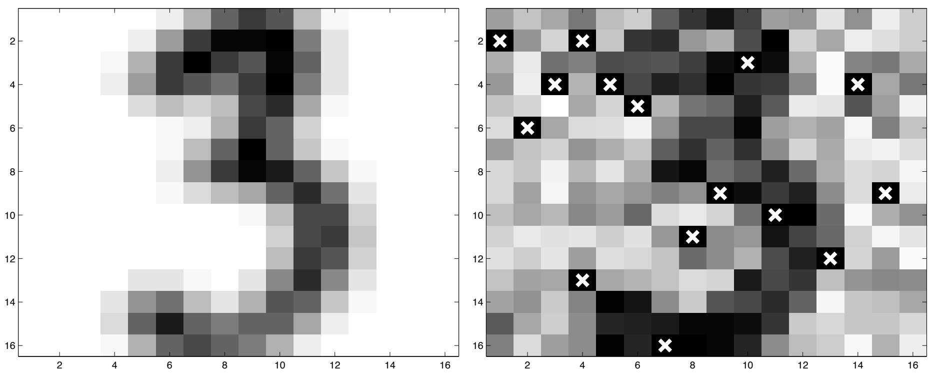

Figure 21.3 Fourth data point from the `USPS` data set. (*Left*) Original image. (*Right*) After rescaling, adding noise, and masking dimensions (x).

COIL The Columbia object image library (COIL-100) is a set of color images of 100 different objects taken from different angles (in steps of 5 degrees) at a resolution of 128×128 pixels (Nene et al., 1996).[1] To create our data set, we first downsampled the red channel of each image to 16×16 pixels by averaging over blocks of 8×8 pixels. We then randomly selected 24 of the 100 objects (with $24 * 360/5 = 1728$ images). The set of 24 objects was partitioned into six classes of four objects each. We then randomly discarded 38 images of each class, to leave 250

1. at `http://www1.cs.columbia.edu/CAVE/research/softlib/coil-100.html`

each. Finally, we applied algorithm 21.1 (with $\sigma = 2$) to hide the image structure from the benchmark participants. Figure 21.4 gives an illustration.

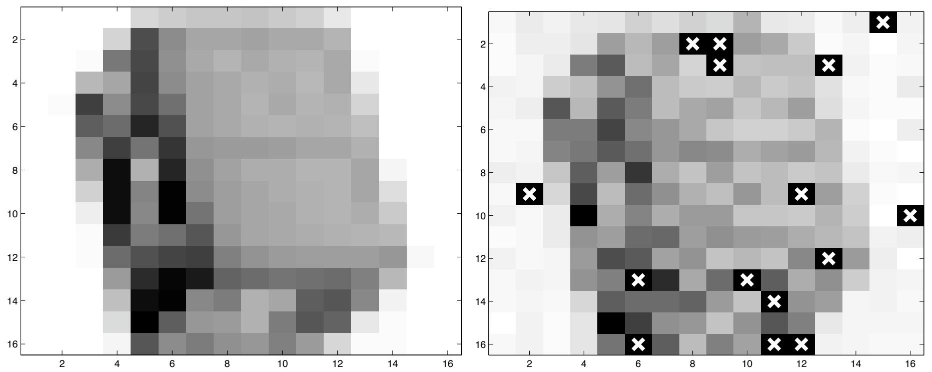

Figure 21.4 First data point from the `COIL` data set. (*Left*) Original image. (*Right*) After rescaling, adding noise, and masking dimensions (x).

BCI This data set originates from research toward the development of a brain computer interface (BCI) (Lal et al., 2004). A single person (subject C) performed 400 trials in each of which he imagined movements with either the left hand (class -1) or the right hand (class +1). In each trial, EEG (electroencephalography) was recorded from 39 electrodes. An autoregressive model of order 3 was fitted to each of the resulting 39 time series. The trail was represented by the total of $117 = 39 * 3$ fitted parameters. We thank Navin Lal for providing these data.

Text This is the 5 `comp.*` groups from the `Newsgroups` data set and the goal is to classify the `ibm` category versus the rest (Tong and Koller, 2001). We are thankful to Simon Tong for providing this data set. A tf-idf (term frequency – inverse document frequency) encoding resulted in a sparse representation with 11,960 dimensions. For the benchmark, 750 points of each class have been randomly selected and the features randomly permuted.

SecStr The main purpose of this benchmark data set is to investigate how far current methods can cope with large-scale application. The task is to predict the secondary structure of a given amino acid in a protein based on a sequence window centered around that amino acid. Our data set is based on the `CB513` set,[2] which was created by Cuff and Barton and consists of 513 proteins (Cuff and Barton, 1999). The 513 proteins consist of a total of 84,119 amino acids, of which 440 were `X`, `Z`, or `B`, and were therefore not considered.

2. e.g. at `http://www.compbio.dundee.ac.uk/~www-jpred/data/pred_res/`

For the remaining 83,679 amino acids, a symmetric sequence window of amino acids [-7,+7] was used to generate the input $\mathbf{x}$. Positions before the beginning or after the end of the protein are represented by a special (21st) letter. Each letter is represented by a sparse binary vector of length 21 such that the position of the single 1 indicates the letter. The 28,968 α-helical and 18,888 β-sheet protein positions were collectively called class -1, while the 35,823 remaining points ("coil") formed class +1.

We supplied another 1,189,472 unlabeled data points. However, none of the benchmark participants chose to utilize these data in their experiments.

21.1.2 Experimental Setup

transductive setting

We decided to carry out the experiments in the transductive setting (cf. chapter 1): the test set coincides with the set of unlabeled points. First, this is most economical in terms of the required amount of data points. Second, this poses the smallest requirements to participating methods. Otherwise it would have been necessary to develop and implement "out-of-sample extensions" (e.g. Bengio et al. (2004b)) for the inherently transductive algorithms. We expect the prediction accuracy on the unlabeled points to be similar to that achieved on out-of-sample points (after having trained on the same sets of labeled and unlabeled points). Recall, however, that transductive methods have to be retrained for every new set of test data, which may be prohibitive in some practical applications. On the other hand, the retraining offers the potential to learn from an increasing amount of unlabeled data, namely the accumulated test points. This potential is wasted when an inductive classifier is trained only once and from there on used.

numbers of labeled points

An important question is how many labeled points are required to achieve decent classification accuracy. To shed some light on this, we equipped the benchmark data sets with subsets of labeled points of different sizes. More precisely, the number of labeled points is either 10 or 100 for all data sets except for `SecStr`, for which it is 100, 1000, or 10,000. In order to make the accuracy estimates derived from the experiments robust and independent of coincidental properties of the chosen points, we devised twelve subsets for each combination of data set and number of labeled points (ten for data set `SecStr`). When choosing the subsets of labeled points, we take care to pick at least one point from each class.

model selection

Since the unions of the sets of labeled points already cover substantial parts of the entire data sets, we provided the labels of all points to the participants of the benchmarks. This allowed for finding hyperparameter values by minimizing the test error, which is not possible in real applications; however, the results of this procedure can be useful to judge the potential of a method. To obtain results that are indicative of real world performance, the model selection has to be performed using only the small set of labeled points.

Table 21.2 TSVM results. For the linear kernel the algorithm described in chapter 6 has been used, for the nonlinear kernel the one of (Chapelle and Zien, 2005).

	`g241c`	`g241d`	`Digit1`	`USPS`	`COIL`	`BCI`	`Text`	
Linear	20.95	46.35	20.59	30.66		50.04	28.60	$n = 10$
Nonlinear	24.71	50.08	17.77	25.20	67.50	49.15	31.21	
Linear	18.18	23.76	18.05	21.12		42.67	22.31	$n = 100$
Nonlinear	18.46	22.42	6.15	9.77	25.80	33.25	24.52	

21.2 Application of SSL Methods

A major problem in the application of SSL methods to problems with very few labeled data points is the model selection. In the following we describe for each method how this was approached. Unless mentioned otherwise, the experiments have been conducted by the authors of the corresponding chapters.

Several authors have provided results corresponding to different variations of their algorithm. In order to keep the final results table as concise as possible, we have in these cases compared them and preselected the best one.

Finally, all the results reported on the tables below are test errors in %.

21.2.1 Transductive Support Vector Machines

Thorsten Joachims has reported results for the transductive support vector machine (TSVM) algorithm described in chapter 6 and for the spectral graph transducer (SGT) (Joachims, 2003). He used the code available on his webpage.

No model selection or parameter tuning has been performed, and according to Joachims the results are likely to be improved by appropriate preprocessing and/or model selection. For TSVM, a linear kernel was used and C was fixed to $C^{-1} = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i||^2$. For the SGT algorithm, the hyperparameters were set as in (Joachims, 2003): $C = 3200$, $d = 80$, and $k = 100$.

Since we believe that on some data sets nonlinearity might be important, we ran our own implementation of TSVM (Chapelle and Zien, 2005) with a radial basis function (RBF) kernel. Its width was chosen as the median of the pairwise distances and C was fixed to 100. Results are presented in table 21.2. The tables at the end of this chapter will refer to the nonlinear version.

21.2.2 Entropy Regularization

The method described in chapter 9 can be kernelized, but the experiments have been reported using a linear classifier. The hyperparameters (λ and weight decay) have been chosen by cross-validation. In case of a tie the smaller λ and the larger weight decay have been selected. Since the algorithm is similar to TSVM (cf. section 21.2.1)

Table 21.3 Performances of the entropy regularization method (cf. chapter 9). Because of the links with TSVM and the use of a linear classifier, the comparison with linear TSVM (see table 21.2) is relevant.

	g241c	g241d	Digit1	USPS	COIL	BCI	Text	
Entropy-Reg.	47.36	45.81	24.44	20.25	66.53	47.71	42.07	$n = 10$
Linear TSVM	20.95	46.35	20.59	30.66		50.04	28.60	
Entropy-Reg.	20.97	25.36	7.28	12.21	29.48	28.89	24.86	$n = 100$
Linear TSVM	18.18	23.76	18.05	21.12		42.67	22.31	

and a linear class of function has been used, we decided to compare the results with those of the linear TSVM. The comparison is shown in table 21.3.

21.2.3 Data-Dependent Regularization

The experiments were run using the distributed propagation data-dependent regularization, which is applicable to both relational data and data derived from a metric. The most important modeling decision in applying data-dependent regularization is the selection of the regions that bias label similarity. In the absence of domain knowledge, k-nearest neighbor regions, centered at each data point as induced by the default Euclidean distance metric, were considered.

In order to determine the number of points in each region tenfold cross-validation experiments were run. For this purpose, points that were graph-disconnected from training data were always treated as errors; this encouraged selecting a k that makes the information regularization graph connected.

The weight of labeled training data against unlabeled data, λ, was set to 0, meaning that the posterior labels of training data were not allowed to change from their given values. The regularization iteration proceeded until the change in parameters became insignificant.

As a result of data-dependent regularization, each previously unlabeled point now had a probabilistic class label. This probabilistic class label was converted to a real label by thresholding the probability. The threshold was applied as an additive term to the log probability of each class. Then the class assigned by the classifier was determined by maximizing the threshold-adjusted (log) probability.

Proper selection of the threshold requires cross-validation. However, for computational efficiency reasons the authors cross-validated only between two scenarios: the first, in which the threshold applied to each class is 0, which corresponds to treating the output of information regularization as plain probabilities; and the second, in which the threshold of each run is optimized so that the resulting class frequency on the unlabeled data matches the empirical class frequency on the labeled observations. Data sets 1, 3, 4, and 6 preferred the first algorithm for selecting the threshold (that is, no threshold), while data sets 2, 5, and 7 preferred the second algorithm.

Table 21.4 Influence of the class mass normalization (CMN, cf. chapter 11)

	g241c	g241d	Digit1	USPS	COIL	BCI	Text	
Without CMN	50.07	49.47	19.66	19.63	61.50	50.66	49.99	$n = 10$
With CMN	39.96	46.55	9.80	13.61	59.63	50.36	40.79	
Without CMN	39.37	36.42	3.17	10.65	10.01	46.92	30.54	$n = 100$
With CMN	22.05	28.20	3.15	6.36	10.03	46.22	25.71	

21.2.4 Label Propagation and Quadratic Criterion

A fully connected graph with an RBF kernel has been chosen for the algorithm described in chapter 11. More precisely, the cost function (11.11) is minimized giving the closed-form solution (11.12). The kernel bandwidth σ was selected in the following way:

- For data sets with 100 labeled examples, by cross-validation on the first split, the same σ being used on all other splits
- For data sets with 10 labeled examples, with the following basic heuristic: $\sigma = d/3$, where d is the estimated average distance between a point in the data set and its 10th nearest neighbor

The tradeoff coefficient μ was set to 10^{-6}.

As shown in table 21.4, the class mass normalization (cf. section 11.5) seemed to be very important, and later results are reported using this technique.

21.2.5 The Manifold Basis of Semi-Supervised Learning

Experiments have been conducted using the semi-supervised kernel introduced in section 12.4. This kernel was used either in an SVM or in regularized least squares (RLS; aka kernel ridge regression).

The kernel is of the form

$$\tilde{K}(x,z) = K(x,z) - \mathbf{k}_x^\top (I + rL^pG)^{-1} L^p \mathbf{k}_z,$$

where $K(x,z)$ is a base kernel, $[\mathbf{k}_x]_i = K(x_i, x)$, G is the Gram matrix (of size $l+u$), L is the normalized graph Laplacian, and r is the ratio $\frac{\gamma_I}{\gamma_A}$.

The base kernel was chosen to be an RBF with width σ. L is computed from an adjacency matrix W corresponding to a weighted k nearest neighbors graph with weights $W_{ij} = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma_{\mathcal{G}}}\right)$ if there is an edge between x_i and x_j, and 0 otherwise. The width $\sigma_{\mathcal{G}}$ is fixed as the mean distance between adjacent nodes on this graph. The adjacency matrix is symmetrized by setting $W_{ij} = W_{ji}$ for any non-zero edge weight W_{ji}. The normalized graph Laplacian is computed as $L = I - D^{-1/2} W D^{-1/2}$ where D is a diagonal degree matrix given by $D_{ii} = \sum_j W_{ij}$.

For all data sets except `Text`, $k = 5, p = 2$ was used. For `Text`, those values are $k = 50, p = 5$. This is based on the experimental experience of the authors: relatively smaller values of k and p tend to work well for image data sets and larger values are useful for textual data sets. No further optimization on these parameters was attempted.

For the multiclass data set, a one-vs.-the-rest strategy was used. For each of the classifiers, the bias b was selected such that a sixth of the unlabeled data was classified in the positive class (because of a prior on uniform class probabilities for the six classes).

The hyperparameters were chosen by performing a search over the following grid:

1. regularization parameter $\gamma_A \in \{10^{-6}, 10^{-4}, 10^{-2}, 1, 100\}$;
2. base kernel width $\sigma \in \{\frac{\sigma_0}{8}, \frac{\sigma_0}{4}, \frac{\sigma_0}{2}, \sigma_0, 2\sigma_0, 4\sigma_0, 8\sigma_0\}$, where σ_0 is the mean norm of the feature vectors in the data set;
3. ratio $r = \frac{\gamma_I}{\gamma_A} \in \{0, 10^{-4}, 10^{-2}, 1, 100, 10^4, 10^6\}$.

For data sets `COIL` and `SecStr`, the best mean test error across splits was reported. For other data sets, the model selection criterion used was either

- fivefold cross-validation error for 100 labeled points, or
- for 10 labeled points, the normalized cut, $\frac{y^\top L^p y}{|i, y_i = 1|\ |i, y_i = -1|}$, where y is the vector of predicted labels.

Data set 8 was treated differently due to its size. The linear Laplacian support vector machine/regularized least squares (SVM/RLS) was run as described in section 12.5 and (Keerthi and DeCoste, 2005). The values $k = 5, p = 4$ were set based on a crude search. Efficient nonlinear methods are currently under development and may possibly return better performance on this data set.

It is important to note that Laplacian SVM/RLS also provides out-of-sample prediction on completely unseen test points. Experimental results on data set `Digit1` are provided in chapter 12 Results are presented on table 21.5. Since LapRLS achieved slightly better performances, we consider this method for the table at the end of this chapter.

21.2.6 Discrete Regularization

This method consists in minimizing (13.12) with $p = 2$ as explained in section 13.3.1. The experiments have been carried out by Mingrui Wu. The value μ was set to 0.05. A k-nearest neighbor graph was constructed with weights on edges (i, j) computed as $\exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$. The values for k and γ were selected by tenfold cross-validation in the sets $\{5, 10, 20, 50, \infty\}$ and $\{\frac{1}{64}, \frac{1}{16}, \frac{1}{4}, 1, 4, 16, 64\}$ respectively. The input data are normalized such that the $\frac{1}{c^2}$ quantile of the pairwise distances equals 1, where c is number of classes.

Table 21.5 Semi-supervised kernel (chapter 12). No MS stands for "no model selection": this is the best mean test error achieved across all hyperparameter values.

	g241c	g241d	Digit1	USPS	COIL	BCI	Text	
LapRLS	43.95	45.68	5.44	18.99	–	48.97	33.68	$n = 10$
LapRLS – no MS	41.74	41.46	6.54	14.67	54.54	46.35	33.35	
LapSVM	46.21	45.15	8.97	19.05	–	49.25	37.28	
LapSVM – no MS	45.53	43.55	6.58	14.99	56.87	46.43	34.04	
LapRLS	24.36	26.46	2.92	4.68	–	31.36	23.57	$n = 100$
LapRLS – no MS	23.45	24.77	1.81	4.31	11.92	27.89	23.32	
LapSVM	23.82	26.36	3.13	4.70	–	32.39	23.86	
LapSVM – no MS	23.43	24.66	2.19	4.36	13.21	28.58	23.08	

21.2.7 Semi-Supervised Learning with Conditional Harmonic Mixing

This method is used to improve the performance of a supervised base classifier. A detailed description of its application to five of the benchmark data sets can be found in chapter 14. In a nutshell, an SVM is trained on the labeled points, and used to predict an initial (delta-) distribution on each unlabeled point. These are used to estimate conditional probability distributions that are associated to the edges of a directed graph with the data points as nodes. The authors took care to make the method essentially free of hyperparameters by averaging over a number of graphs constructed in different ways, although they conclude from their experiments that clever model selection might be able to perform better (cf. chapter 14).

21.2.8 Spectral Methods for Dimensionality Reduction

For the dimensionality methods described in chapter 16, a number k of nearest neighbors has to be chosen and the manifold dimensionality has to be estimated. k was set 3 for maximum variance unfolding (MVU), 12 for locally linear embedding (LLE), and 6 for Isomap and Laplacian eigenmaps. The dimensionality was estimated such that MVU explains 99% of the variance of the data (cf. table 21.6). After dimensionality reduction, the 1-nearest neighbor algorithm was used.

Table 21.6 First line: number of components kept in the dimensionality reduction; second line: "true" manifold dimension; third line: estimate of the manifold dimension according to the method described in (Hein and Audibert, 2005). [3]

g241c	g241d	Digit1	USPS	COIL	BCI	Text
38	33	4	9	3	8	29
241	241	5	?	1	?	?
66	63	15	4	2	9	7

Table 21.7 Nonlinear dimensionality reduction (chapter 16)

	g241c	g241d	Digit1	USPS	COIL	BCI	Text	
Isomap	47.88	46.72	13.65	16.66	63.36	49.00	38.12	
LapEig	47.47	45.34	12.04	19.14	67.96	49.94	40.84	
LLE	47.15	45.56	14.42	23.34	62.62	47.95	45.32	
MVU	48.68	47.28	11.92	14.88	65.72	50.24	39.40	$n = 10$
PCA	39.38	37.03	21.70	23.40	67.88	49.17	41.65	
None	44.05	43.22	23.47	19.82	65.91	48.74	39.44	
Isomap	43.93	42.45	3.89	5.81	17.35	48.67	30.11	
LapEig	42.14	39.43	2.52	6.09	36.49	48.64	30.92	
LLE	43.01	38.20	2.83	6.50	28.71	47.89	32.83	
MVU	44.05	43.21	3.99	6.09	32.27	47.42	30.74	$n = 100$
PCA	33.51	25.92	8.27	9.50	28.41	48.58	28.83	
None	40.28	37.49	6.12	7.64	23.27	44.83	30.77	

The performances of the different dimensionality reduction methods can be found in table 21.7. Note that principal components analysis (PCA) can achieve a very good performance on some data sets; for instance, with 100 labeled points, the test error is 17.3% on g241c, 9% on g241d, and 27.7% on Text, if, respectively, 1, 3, and 20 components are chosen. For the first two data sets, this is not really surprising given the artificial nature of the data. For Text, this can be explained by the fact that PCA performs *latent semantic analysis* (Deerwester et al., 1990). Finally, note that additional dimensions would have been helpful for the COIL data set. Indeed, with 12 components, Isomap achieve a test error of 12% (for 100 labeled points).

21.2.9 Large-Scale Algorithms

The large-scale methods described in chapter 18 use a small set of size m on which to expand the solution. m was fixed to 100, except for the large-scale data set SecStr where m was set to 1000.

The length scale σ was selected as explained in section 21.2.4, except that for ten labeled points, the distance d used in the heuristic $\sigma = d/3$ is calculated as the average distance between a point and its 10th nearest neighbor among $m+10$ other points randomly selected.

3. We thank Matthias Hein for having computed those estimates.

Table 21.8 Large-scale strategies (chapter 18)

	`g241c`	`g241d`	`Digit1`	`USPS`	`COIL`	`BCI`	`Text`	
NoSub	39.96	46.55	9.80	13.61	59.63	50.36	40.79	$n = 10$
RandSub	40.11	41.93	15.21	15.64	65.11	49.96	37.37	
SmartSub	39.56	42.20	14.19	18.56	65.94	48.31	38.60	
SmartOnly	39.82	42.24	12.60	16.95	63.97	49.47	38.23	
NoSub	22.05	28.20	3.15	6.36	10.03	46.22	25.71	$n = 100$
RandSub	23.60	25.85	4.20	7.97	19.74	44.61	25.60	
SmartSub	22.07	26.16	4.11	7.51	22.86	44.36	25.71	
SmartOnly	22.07	25.98	3.50	6.90	15.70	44.78	25.75	

Table 21.8 presents results for the following algorithms:

NoSub: No subsampling, i.e. the results of section 21.2.4.

RandSub: Random subsampling.

SmartSub: The method described in algorithm 18.1.

SmartOnly: Training using only a subset of the data selected by algorithm 18.1. This is to be able to assess the usefulness of actually using the rest of the data in the cost (cf. matrix C_{RS} in Eq. 18.7).

21.2.10 Cluster Kernels

The kernel proposed in chapter 19 is a product of two kernels:

1. k_{orig} is a standard RBF kernel with width σ and ridge C^{-1}. Those two hyperparameters have been optimized with the code available at `http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/ams/`.
2. k_{bag} resulting from repeated runs of the k-means algorithm. k has been found by tenfold cross-validation in the set $\{2, 4, 6, 8, 10, 20, 30, 40, 50\}$, the kernel k_{orig} being fixed.

Finally, the method mentioned in footnote 2 in chapter 19 was used with $\lambda = 0.5$ on data sets `COIL` and `SecStr` because it worked better.

21.2.11 Low-Density Separation

This method is not described in the book, but in (Chapelle and Zien, 2005). The code used to run the experiments is available at `http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/lds/`. The hyperparameter ρ is found by cross-validation, the other hyperparameters being fixed to their default values. The reason for not optimizing on more hyperparameters is that the the model selection becomes unreliable, especially with only ten labeled points. Note that if the number k of nearest neighbors is optimized on the test error, the test error can be dramatically decreased: for instance, on `Digit1` with ten labeled points, a test error of 3.7% was achieved with $k = 5$. This has to be compared to the 15.6% achieved by cross-validation on ρ only.

21.2.12 Boosting

Ayhan Demiriz ran experiments on data set `SecStr` using the assemble algorithm (Bennett et al., 2002), which is a modified version of AdaBoost for semi-supervised learning. It turns out that the algorithm was not very well suited for a very small number of labeled points, as the algorithm stops whenever a weak learner correctly classifies all labeled points. On the other hand, it seems much better suited for large data sets, because the run time increases only linearly in the number of labeled and unlabeled points.

The weak learner was a two-level decision tree. AdaBoost and Assemble have both been run for 50 iterations. In this case, it seems that semi-supervised learning was not helpful: AdaBoost achieved 30.8% test error, while Assemble achieved 32.2%.

21.3 Results and Discussion

To compare the results of the different methods, we summarize them in tables. Tables 21.9 and 21.10 show the mean test errors and the ROC (receiver operating characteristic) scores for training with 10 labeled points; similarly tables 21.11 and 21.12 for 100 labeled points. The results for `SecStr` are presented separately in table 21.13 since the numbers of labeled points differ from the other data sets. Further, only a small number of methods competed in this benchmark.

Tables 21.9 and 21.11 contain a lot of results and might be a bit difficult to parse. For this reason, we propose to perform some clustering on the results. Concerning the data sets, we can identify two main categories:

manifold-like data sets

Manifold-like: The data lie near a low-dimensional manifold. Based on table 21.6, this seems to be the case of data sets `Digit1`, `USPS`, `COIL`, and `BCI`. For the first three, this can be easily explained by the fact the data represent images; for `BCI`, this is less obvious, but it seems plausible that the signals captured by an EEG have rather few degrees of freedom.

cluster-like data sets

Cluster-like: The data are clustered, and they tend cluster in such a way that two classes do not share the same cluster. By construction this is the case for data sets `g241c` and `g241d`. We conjecture that `Text` belongs also to this category, because cluster-based algorithms (see below) usually perform well on text data. As for the algorithms, we can also identify two categories, which correspond to the two types

Table 21.9 Test errors (%) with 10 labeled training points. Values printed in italics were obtained by performing model selection w.r.t. the test error.

		g241c	**g241d**	**Digit1**	**USPS**	**COIL**	**BCI**	**Text**
	1-NN	44.05	43.22	23.47	19.82	65.91	48.74	39.44
	SVM	47.32	46.66	30.60	20.03	68.36	49.85	45.37
21.2.8	**MVU + 1-NN**	48.68	47.28	11.92	14.88	65.72	50.24	39.40
21.2.8	**LEM + 1-NN**	47.47	45.34	12.04	19.14	67.96	49.94	40.84
21.2.4	**QC + CMN**	39.96	46.55	9.80	13.61	59.63	50.36	40.79
21.2.6	**Discrete Reg.**	49.59	49.05	12.64	16.07	63.38	49.51	40.37
21.2.1	**TSVM**	24.71	50.08	17.77	25.20	67.50	49.15	31.21
21.2.1	**SGT**	22.76	18.64	8.92	25.36	–	49.59	29.02
21.2.10	**Cluster-Kernel**	48.28	42.05	18.73	19.41	67.32	48.31	42.72
21.2.3	**Data-Dep. Reg.**	41.25	45.89	12.49	17.96	63.65	50.21	–
21.2.11	**LDS**	28.85	50.63	15.63	17.57	61.90	49.27	27.15
21.2.5	**Laplacian RLS**	43.95	45.68	5.44	18.99	*54.54*	48.97	33.68
21.2.7	**CHM (normed)**	39.03	43.01	14.86	20.53	–	46.90	–

Table 21.10 ROC scores (area under curve; %) with 10 labeled training points.

		g241c	**g241d**	**Digit1**	**USPS**	**BCI**	**Text**
	1-NN	–	–	–	–	–	–
	SVM	64.68	63.04	88.38	75.56	51.59	67.97
21.2.8	**MVU + 1-NN**	–	–	–	–	–	–
21.2.8	**LEM + 1-NN**	–	–	–	–	–	–
21.2.4	**QC + CMN**	64.24	62.45	96.32	90.76	49.47	70.71
21.2.6	**Discrete Reg.**	51.75	52.73	91.03	80.65	51.45	53.79
21.2.1	**TSVM**	82.41	50.65	86.98	68.21	50.92	73.42
21.2.1	**SGT**	87.41	89.40	97.58	73.08	50.70	80.09
21.2.10	**Cluster-Kernel**	61.63	77.68	89.49	74.28	51.77	73.09
21.2.3	**Data-Dep. Reg.**	63.43	56.92	96.22	84.91	50.31	–
21.2.11	**LDS**	77.35	49.70	90.10	75.88	49.75	80.68
21.2.5	**Laplacian RLS**	59.23	57.07	99.50	85.70	51.69	76.55
21.2.7	**CHM (normed)**	64.83	62.29	92.91	81.16	52.75	–

of data sets mentioned above:

manifold-based algorithms

Manifold-based: These algorithms come from parts III and IV of this book: `Discrete Reg`, `QC`, `Laplacian RLS`, `CHM`, `SDE`, `LEM`, `SGT`. Note in particular that `Discrete Reg` and `QC` minimize a similar cost function, the difference being the normalization of the Laplacian.

cluster-based algorithms

Cluster-based (or low-density separation as explained in part II of the book: There are three algorithms in this category which are expected to behave similarly: `TSVM`, `Data-Dep Reg`, `Entropy-Reg` (see section 21.2.2). Finally, `Cluster-Kernel` and `LDS` also belong to this category, but are not closely related to the former three.

A first conclusion that we can draw from these experiments is that no algorithm

Table 21.11 Test errors (%) with 100 labeled training points. Values printed in italics were obtained by performing model selection w.r.t. the test error.

		g241c	**g241d**	**Digit1**	**USPS**	**COIL**	**BCI**	**Text**
	1-NN	40.28	37.49	6.12	7.64	23.27	44.83	30.77
	SVM	23.11	24.64	5.53	9.75	22.93	34.31	26.45
21.2.8	**MVU + 1-NN**	44.05	43.21	3.99	6.09	32.27	47.42	30.74
21.2.8	**LEM + 1-NN**	42.14	39.43	2.52	6.09	36.49	48.64	30.92
21.2.4	**QC + CMN**	22.05	28.20	3.15	6.36	10.03	46.22	25.71
21.2.6	**Discrete Reg.**	43.65	41.65	2.77	4.68	9.61	47.67	24.00
21.2.1	**TSVM**	18.46	22.42	6.15	9.77	25.80	33.25	24.52
21.2.1	**SGT**	17.41	9.11	2.61	6.80	–	45.03	23.09
21.2.10	**Cluster-Kernel**	13.49	4.95	3.79	9.68	21.99	35.17	24.38
21.2.3	**Data-Dep. Reg.**	20.31	32.82	2.44	5.10	11.46	47.47	–
21.2.11	**LDS**	18.04	23.74	3.46	4.96	13.72	43.97	23.15
21.2.5	**Laplacian RLS**	24.36	26.46	2.92	4.68	*11.92*	31.36	23.57
21.2.7	**CHM (normed)**	24.82	25.67	3.79	7.65	–	36.03	–

Table 21.12 ROC scores (area under curve; %) with 100 labeled training points.

		g241c	**g241d**	**Digit1**	**USPS**	**BCI**	**Text**
	1-NN	–	–	–	–	–	–
	SVM	85.57	83.54	99.09	95.76	71.17	84.26
21.2.8	**MVU + 1-NN**	–	–	–	–	–	–
21.2.8	**LEM + 1-NN**	–	–	–	–	–	–
21.2.4	**QC + CMN**	86.40	82.23	99.59	91.11	56.48	84.62
21.2.6	**Discrete Reg.**	52.81	52.97	98.84	92.24	52.36	71.53
21.2.1	**TSVM**	88.55	84.18	98.02	92.74	73.09	80.96
21.2.1	**SGT**	91.74	97.48	99.76	96.72	56.79	85.22
21.2.10	**Cluster-Kernel**	93.15	98.95	99.36	94.50	70.50	85.90
21.2.3	**Data-Dep. Reg.**	87.50	74.18	99.81	97.74	54.38	–
21.2.11	**LDS**	89.37	83.13	99.23	95.62	57.22	84.77
21.2.5	**Laplacian RLS**	83.54	81.54	99.40	98.65	74.83	85.05
21.2.7	**CHM (normed)**	81.13	81.36	99.49	96.69	66.32	–

is uniformly better than the others, and that for a given semi-supervised learning problem, the algorithm needs to be selected carefully as a function of the nature of the data set. A general rule (which seems obvious a posteriori) is that manifold-based algorithms should be used for manifold-like data sets, and cluster-based algorithms should be used for cluster-like data sets.

model selection

It should be also noted that model selection was challenging for most of the competitors, especially in the case of only 10 labeled points, where the use of cross-validation can be unreliable. In this respect, the results with 100 labeled points are expected to be more reliable and to give a better indication of the strength of the different algorithms.

limits of semi-supervised learning

One of the disappointing results of this benchmark is the `Text` data set. Indeed, it has been shown that semi-supervised learning can be very useful for this type

Table 21.13 Results for SecStr for different numbers of labeled points. (*Left*) Test error (%). (*Right*) ROC score (%). Values printed in italics were obtained by performing model selection w.r.t. the test error.

	100	**1000**	**10000**	**100**	**1000**	**10000**
SVM	44.59	33.71	–	59.09	70.86	–
Cluster Kernel	42.95	34.03	–	58.79	70.37	–
QC randsub (CMN)	42.32	40.84	–	54.77	59.99	–
QC smartonly (CMN)	42.14	40.71	–	55.59	60.25	–
QC smartsub (CMN)	42.26	40.84	–	55.35	60.08	–
Boosting (Assemble)	–	–	32.21	–	–	–
LapRLS	*42.59*	*34.17*	*28.55*	*59.02*	*70.33*	*77.95*
LapSVM	*43.42*	*33.96*	*28.53*	*58.40*	*70.54*	*77.95*

of data (Joachims, 1999; Nigam et al., 2000; Chapelle and Zien, 2005), but the results from tables 21.9 and 21.11 exhibit only a moderate improvement over plain supervised learning. The fact that the data set has been constructed in a one-vs-rest setting could be a possible explanation (cf. section 21.1.1). To test this hypothesis we tried to classify only two topics, namely ibm and x. A linear SVM achieved a mean test error of 12% (over several subsets of 100 labeled points), while a linear TSVM was able to reduce the test error to 2%. Further investigation is required to understand why such a large improvement is possible in this case.

Finally, it is worth pointing out that one should not necessarily expect an improvement with unlabeled data. The data sets BCI and SecStr seem to be examples where it is difficult to do better than standard supervised learning. At least for SecStr, this might be a problem of the amounts of unlabeled data that are utilized. Current approaches to protein secondary structure prediction use essentially all known protein sequences, which amount to tens or hundreds of million unlabeled data points. This is only possible due to the use of a very simple strategy: roughly speaking, each protein is represented by an average of the proteins in its neighborhood (Rost and Sander, 1993). Clearly, bringing the more sophisticated (and probably more powerful) SSL methods to this scale is an important open problem.

In all cases, we believe that there is no "black box" solution and that a good understanding of the nature of the data is required to perform successful semi-supervised learning. Indeed, in supervised learning, it seems that a good generic learning algorithm can perform well on a lot of real-world data sets without specific domain knowledge. In contrast, semi-supervised learning is possible only due to the special form of the data distribution that correlates the label of a data point with its situation within the distribution; therefore it seems much more difficult to design a general semi-supervised classifier. Instead, powerful semi-supervised learning algorithms distinguish themselves through the ability to make use of available prior knowledge about the domain and data distribution, in order to relate data and labels and improve classification. [4]

4. Part of this paragraph has been inspired by comments from Adrian Corduneanu.